Theoretical Computer Science 23 (1983) 287–304
North-Holland Publishing Company

REAL FUNCTIONS AND NUMBERS DEFINED BY TURING MACHINES

Rudolf FREUND

Institut für Mathematische Logik und Formale Sprachen, Technische Universität Wien, Wien, Austria

Communicated by M. Nivat
Received April 1981
Revised April 1982

Abstract. Real functions and real numbers defined by deterministic or non-deterministic Turing machines are observed, using the interpretation of infinite strings of digits as real numbers. A hierarchy of such Turing functions and numbers is established, and two classes of transcendent Turing numbers are investigated.

1. Preliminaries

The notation concerning ω-languages is mainly taken from [4] and [5] as well as from [6], where the basic results about the interpretation of ω-words as real numbers are to be found.

An *alphabet* V is a finite non-empty set of *symbols*. A finite string (*word*) over V is any sequence $x = \prod_{i=1}^{k} a_i$, $a_i \in V$, $i = 1, \ldots, k$, $k = 0, 1, \ldots$; k is the *length* of x, and for each $a \in V$, $n_a(x)$ is the number of symbols a in x. V^* denotes the set of all words over V, $V^+ = V^* - \{\lambda\}$, where λ denotes the empty word.

Let N denote the set of natural numbers, and let $N_0 = N \cup \{0\}$ be the set of non-negative integers. For $k \geq 2$, the set of non-negative integers $< k$ shall be denoted by N_k.

If x is a real number, $|x|$ is the *absolute value* of x. $[a, b]$ is the closed interval of real numbers $a \leq x \leq b$, (a, b) the open interval of real numbers $a < x < b$. The set of rational numbers in $[0, 1]$ shall be denoted by Q.

The *domain* of a mapping (function) f will be abbreviated by $\mathrm{dom}(f)$. A real function $f: [0, 1] \to [0, 1]$ is called *continuous* on the interval $[a, b] \subseteq \mathrm{dom}(f)$, if for any $\varepsilon \in (0, 1)$ there is a $\delta \in (0, 1)$ so that for all $x, y \in [a, b]$ $|x - y| < \delta$ implies $|f(x) - f(y)| < \varepsilon$.

Definition 1. For any alphabet V, let V^ω denote the set of all infinite strings $x = \prod_{i=1}^{\infty} a_i$, $a_i \in V$, $i \in N$, over V; x is called an *ω-word*. Any subset of V^ω is an *ω-language*. For any language $L \subseteq V^*$ an ω-language L^ω can be defined consisting

0304-3975/83/0000-0000/$03.00

of all ω-words obtained by concatenating words of L in an infinite sequence:

$$L^{\omega} = \left\{ x \in V^{\omega} : x = \prod_{i=1}^{\infty} x_i, \text{ where for each } i \in N, x_i \in L - \{\lambda\} \right\}.$$

For any ω-word $x = \prod_{i=1}^{\infty} a_i$, $a_i \in V$, $i \in N$, define for each $j \in N$,

$$x/j = \prod_{i=1}^{j} a_i, \qquad x(j) = a_j \quad \text{and} \quad x/0 = \lambda.$$

For any finite string $x = \prod_{i=1}^{k} a_i$, $k \in N_0$, the notations x/j and $x(j)$, $0 \leq j \leq k$, shall be used, too.

V^{ω} is not a monoid like V^*; however, given $s = \prod_{i=1}^{k} a_i$ and $t = \prod_{j=1}^{\infty} b_j$, $k \in N_0$, $a_i, b_j \in V$, the *product* $s \cdot t = st = \prod_{i=1}^{k} a_i \prod_{j=1}^{\infty} b_j = a_1 \cdots a_k b_1 b_2 \cdots$ can be defined in the usual way.

2. Basic definitions

As throughout this paper the analysing and generating of ω-words by Turing machines is studied, a precise definition of the model for these machines chosen in order to be able to present simple and clear examples, is given in the following lines.

Definition 2. A *Turing machine* is a 6-tuple $T = (K, \Sigma, \Gamma_Z, Z, \delta, q_0)$, where K is a finite set of states, $Z \in \Gamma_Z$ is the left boundary marker, Σ is the input alphabet, Γ_Z is the tape alphabet so that $\Sigma \subseteq \Gamma$ and $\Gamma = \Gamma_Z - \{Z\}$, $q_0 \in K$ is the initial state, and δ is a mapping from $K \times \Gamma$ to subsets of $K \times \Gamma \times \{L, R\}$.

A *configuration* of T is an ω-word in $Z\Gamma^* K \Gamma^{\omega} \cup K Z \Gamma^{\omega}$. The relation $\vdash_T$ on the set of all configurations is defined as follows: $x \vdash_T y$ iff

(i) $x = pZw$, $y = Zqw$ and $(q, Z, R) \in \delta(p, Z)$ or

(ii) $x = uApBw$, $y = uqACw$ and $(q, C, L) \in \delta(p, B)$ or

(iii) $x = vpBw$, $y = vCqw$ and $(q, C, R) \in \delta(p, B)$,

where $p, q \in K$, $w \in \Gamma^{\omega}$, $A \in \Gamma_Z$, $uA \in Z\Gamma^*$, $B, C \in \Gamma$, and $v \in Z\Gamma^*$.

T is called *deterministic*, if δ is a mapping from $K \times \Gamma$ to $K \times \Gamma \times \{L, R\}$ only.

In the classical theory one regards the reflexive and transitive closure of $\vdash_T$, $\vdash_T^*$, and the actions of the Turing machine on ω-words Zxb^{ω}, where x is an element of Σ^* and b is a special letter of $\Gamma - \Sigma$, called the blank symbol. The language accepted by T is $L(T) = \{w \in \Sigma^* :$ there are $u \in \Gamma^*$, $v \in \Gamma^{\omega}$, $q \in F$ so that $Zq_0wb^{\omega} \vdash_T^* uqv\}$, where $F \subseteq K$ is a set of final states. In the theory of ω-languages the actions of the Turing machine T on the semi-infinite tape Zw, $w \in \Sigma^{\omega}$, are regarded.

Definition 3. Let $T = (K, \Sigma, \Gamma_Z, Z, \delta, q_0)$ be a Turing machine, $w \in \Sigma^\omega$. An infinite sequence of configurations $r = (c_1, c_2, \dots)$ is called a *run* of T on w, if $c_1 = Zq_0w$ and for each $i \in N$, $c_i \vdash_T c_{i+1}$. Run r is called *complete*, if during the infinite computation of T on w the Turing machine reaches each square on the tape, i.e. for any $i \geq 2$ there exists an $l \geq 1$ so that $c_l(i) \in K$. A complete run r is called *non-oscillating*, if T scans each square on the tape only finitely many times, i.e. for each $i \geq 1$ there is an $l \geq 1$ so that $c_k/i \in Z\Gamma^*$ for all $k \geq l$; otherwise r is an *oscillating run*.

A computation which does not correspond to a complete run may either be finite, in case the machine halts on w, or else be an infinite oscillating run where the reading head never leaves an initial segment x/j for some $j \geq 2$.

If there is a complete non-oscillating run r of T on w, then by definition an ω-word $x \in \Gamma^\omega$ must exist so that for each $i \geq 2$ there is an $l \geq 1$ with $c_k/i = Zx/i$ for all $k \geq l$; x is well defined, because for each complete non-oscillating run r the corresponding ω-word x is unambiguously determined by the above condition, and we shall write $Zq_0w \vdash_T^\omega Zx$. The *ω-language accepted by T* $L_\omega(T)$ is the set of all ω-words leading to a complete non-oscillating run, i.e.

$$L_\omega(T) = \left\{ w \in \Sigma^\omega : Zq_0w \overset{\omega}{\underset{T}{\vdash}} Zx \text{ for an } \omega\text{-word } x \in \Gamma^\omega \right\}.$$

We now shall restrict our attention to the alphabets $\Sigma = N_k$, because the following definition allows to regard the elements of N_k^ω as the k-adic representations of the real numbers in the interval [0, 1] (see [6, p. 363]).

Definition 4. For any integer $k \geq 2$ the function $\mu_k : N_k^\omega \to [0, 1]$ is defined by the formula $\mu_k(s) = \sum_{i=1}^{\infty} s(i)/k^i$ for all $s \in N_k^\omega$.

If $x \in [0, 1]$, $x^{(k)} \in N_k^\omega$, and $x = \mu_k(x^{(k)})$, then x is called the *interpretation* of $x^{(k)}$ and $x^{(k)}$ is called an *expansion* of x at the base k.

Theorem 1 (see [6, p. 364]). *Each real number $0 \leq x \leq 1$ has an expansion $x^{(k)}$, $x = 0 . x^{(k)}(1)x^{(k)}(2) \dots$, which is unique, except when $x = u/k^n$ with $u, n \in N$, $0 < u < k^n$, u not divisible by k; in this case there are exactly two expansions $x = 0 . sd0^\omega$ and $x = 0 . s(d-1)(k-1)^\omega$ with $s \in N_k^*$, $d \in N_k$, $d \neq 0$, $|sd| = n$, $\mu_k(sd0^\omega)k^n = u$; $x = 0 . sd0^\omega$ is called the* terminating expansion, *while $x = 0 . s(d-1)(k-1)^\omega$ is called the* non-terminating expansion.

Let $Q(k)$ denote the set of all rational expansions uv^ω, $u \in N_k^*$, $v \in N_k^+$; *then, for any $s \in N_k^\omega$, the following statement obviously holds true:*

$$s \in Q(k) \quad \textit{iff} \quad \mu_k(s) \in Q.$$

To consider functions $f : N_k^\omega \to N_k^\omega$ as real functions, the ambiguity of the expansions $x^{(k)} \in Q(k)$ of some elements $x \in Q$ requires the following notion:

Definition 5. A function $f: N_k^\omega \to N_k^\omega$ is *consistent*, if $s_1, s_2 \in N_k^\omega$, $\mu_k(s_1) = \mu_k(s_2)$ implies $\mu_k(f(s_1)) = \mu_k(f(s_2))$. The function f, if being consistent, induces a real function $f^r: [0, 1] \to [0, 1]$ with $\mu_k(f(s)) = f^r(\mu_k(s))$ for all $s \in \text{dom}(f)$ and $\text{dom}(f^r) = \mu_k(\text{dom}(f))$.

If $T = (K, \Sigma, \Gamma_Z, Z, \delta, q_0)$ is a deterministic Turing machine, then for each $w \in \Sigma^\omega$ exactly one run of T on w is possible, and therefore at most one $x \in \Gamma^\omega$ may exist so that $Zq_0w \vdash_T^\omega Zx$; in this case we shall also use the notation $T(w) = x$, and T may be regarded as a mapping from Σ^ω to Γ^ω. Yet we shall restrict our attention to functions from Σ^ω to Σ^ω only. A deterministic Turing machine T therefore induces a mapping $T_\Sigma: \Sigma^\omega \to \Sigma^\omega$, where $T_\Sigma(w) = T(w)$, if $T(w)$ is an element of Σ^ω, and $T_\Sigma(w)$ remains undefined otherwise. If T is a non-deterministic Turing machine, the following condition may hold true:

Condition 1. For any $w \in \Sigma^\omega$ there exists at most one ω-word $x \in \Sigma^\omega$ so that $Zq_0w \vdash_T^\omega Zx$.

Then the non-deterministic Turing machine T induces a function $T_\Sigma: \Sigma^\omega \to \Sigma^\omega$, too, $T_\Sigma(w)$ being defined as above for deterministic Turing machines.

In [9] special deterministic Turing machines are proposed, and we shall call a deterministic Turing machine *strongly deterministic*, if the following condition holds true:

Condition 2. For any $w \in \Sigma^\omega$ there exists a complete non-oscillating run $r = (c_1, c_2, \ldots)$ of T on w so that $c_i/n = Zv$, $n \in N$, $v \in \Sigma^+$, implies $c_j/n = Zv$ for all $j > i$.

As will be seen in the following sections, strongly deterministic Turing machines may carry out rather complicated computations. Yet if we replaced '$v \in \Sigma^+$' by '$v \in \Gamma^+$' in Condition 2, the resulting devices would correspond with 'sequential machines' ('Mealy machines') as they are defined in [6], because only infinite runs $r = (c_1, c_2, \ldots)$ with $c_i(i+1) \in K$ for all $i \in N$ were possible. Remarkable examples of (real) functions induced by sequential machines are given in [6, pp. 370–377].

The family of functions $T_\Sigma: \Sigma^\omega \to \Sigma^\omega$ induced by arbitrary (resp. deterministic, strongly deterministic) Turing machines shall be denoted by $\text{TF}(\Sigma)$ ($\text{DTF}(\Sigma)$, $\text{SDTF}(\Sigma)$). If $\Sigma = N_k$ for some $k \geq 2$, any consistent function T_Σ in $\text{TF}(N_k)$ ($\text{DTF}(N_k)$, $\text{SDTF}(N_k)$) induces a real function $T_{N_k}^r: [0, 1] \to [0, 1]$. Therefore a real function $f: [0, 1] \to [0, 1]$ will be called a *Turing function* (resp. *deterministic Turing function*, *strongly deterministic Turing function*), if $f = T_{N_k}^r$ for some $k \geq 2$ and an arbitrary consistent function T_{N_k} in $\text{TF}(N_k)$ ($\text{DTF}(N_k)$, $\text{SDTF}(N_k)$). Any constant function $T_{N_k}^r(x) = c$ for all $x \in [0, 1]$ defines a real number $c \in [0, 1]$, and the resulting families of *Turing numbers* – corresponding to the families of *Turing functions* TF, DTF, and SDTF – are denoted by TN, DTN, and SDTN.

For example, any $x \in Q(k)$, $k \geq 2$, is computable by a strongly deterministic Turing machine, hence $Q \subseteq \mathrm{SDTN}$. As the results of Section 3 will show, this inclusion is proper. The following examples will illustrate some other definitions given in this section.

Example 1. The identity function $i:[0, 1] \to [0, 1]$ is an element of SDTF. In fact, for any $k \geq 2$, the strongly deterministic Turing machine $I^{(k)} = (\{q_0\}, N_k, N_k \cup \{Z\}, Z, \delta, q_0)$ with $\delta(q_0, a) = (q_0, a, R)$ for all $a \in N_k$ induces a consistent function $I^{(k)}_{N_k}: N_k^\omega \to N_k^\omega$ so that $I^{(k)r}_{N_k} = i$.

Example 2. The strongly deterministic Turing machine $T' = (\{q_0, r_0, r_1\}, N_2, N_2 \cup \{Z\}, Z, \delta, q_0)$ with $\delta(q_0, 0) = (r_0, 0, R)$, $\delta(r_0, 0) = \delta(r_0, 1) = (r_0, 0, R)$, and $\delta(q_0, 1) = (r_1, 1, R)$, $\delta(r_1, 0) = \delta(r_1, 1) = (r_1, 1, R)$ yields a function $T'_{N_2}: N_2^\omega \to N_2^\omega$ with $\mathrm{dom}(T'_{N_2}) = N_2^\omega$ and $T'_{N_2}(0w) = 0^\omega$, $T'_{N_2}(1w) = 1^\omega$ for all $w \in N_2^\omega$. T'_{N_2} is not consistent, because in the point $\frac{1}{2} \in Q$ the two expansions of $\frac{1}{2}$ in $Q(2)$, 01^ω and 10^ω, have different values.

As will be shown in the following section, no strongly deterministic Turing machine T with $\mathrm{dom}(T^r_{N_2}) = [0, 1]$, $T^r_{N_2}(x) = 0$ for all $x < \frac{1}{2}$ and $T^r_{N_2}(x) = 1$ for all $x > \frac{1}{2}$ can exist, but in the following example a deterministic Turing machine, having this property, is constructed.

Example 3. Let $T = (\{q_0, r_0, r_1, l, p\}, N_2, N_2 \cup \{Z\}, Z, \delta, q_0)$ be the deterministic Turing machine with $\delta(q_0, 0) = (r_0, 0, R)$, $\delta(r_0, 0) = \delta(r_0, 1) = (r_0, 0, R)$, $\delta(q_0, 1) = (p, 0, R)$, $\delta(p, 0) = (p, 0, R)$, $\delta(p, 1) = (l, 1, L)$, $\delta(l, 0) = (l, 0, L)$ $\delta(l, Z) = (r_1, Z, R)$, and $\delta(r_1, 0) = \delta(r_1, 1) = (r_1, 1, R)$. Then $T_{N_2}(0w) = 0^\omega$ for all $w \in N_2^\omega$, $T_{N_2}(10^\omega) = T_{N_2}(01^\omega) = 0^\omega$, and $T_{N_2}(1w) = 1^\omega$ for all $w \in N_2^\omega$, $w \neq 0^\omega$. Therefore T_{N_2} is a consistent function with $\mathrm{dom}(T_{N_2}) = N_2^\omega$, and $T^r_{N_2}(x) = 0$ for all $x \leq \frac{1}{2}$ and $T^r_{N_2}(x) = 1$ for all $x > \frac{1}{2}$.

3. A hierarchy of Turing functions and numbers

In this section, the choice of the notions SDTF, DTF, and TF without specifying the underlying alphabet N_k of digits is justified, and the hierarchy $\mathrm{SDTN} \subsetneq \mathrm{DTN} \subsetneq \mathrm{TN}$, which immediately implies the hierarchy $\mathrm{SDTF} \subsetneq \mathrm{DTF} \subsetneq \mathrm{TF}$, is established. Any function of SDTF is proved to be continuous on closed intervals, which result yields rather simple examples of functions in DTF-SDTF (see e.g. Example 3). Finally, all the three families of Turing functions are proved to be closed under composition.

First a very important property of any kind of Turing machines is shown. A Turing machine can be provided with enough work-space without losing any information initially written on the tape. These actions shall be called the *folding programme*, according to a rather similar process used in [4] and [5].

Lemma 1 (Folding programme). *Any arbitrary (deterministic, strongly deterministic) Turing machine $T = (K, \Sigma, \Gamma_Z, Z, \delta, q_0)$ can be provided with work-space whatever needed during an arbitrary run without losing any information written on the tape at the beginning of the run.*

Proof. During a run of T on an ω-word $w = \prod_{i=1}^{\infty} a_i$, $a_i \in \Sigma$, T may need work-space for computations as are possible in the finite case (where T starts on an ω-word vb^{ω}, $v \in \Sigma^*$, the infinite right tail b^{ω} of the tape providing T with any work-space ever needed); in order not to lose information about any symbol of w, the initial segment $a_1 \cdots a_k$, $k \in N$, may be 'folded forwards' as is shown in the following figure:

Z	a_1	$\cdots$	a_k	a_{k+1}	$\cdots$	a_{2k}	a_{2k+1}	$\cdots$

Z	X_1	$\cdots$	X_k	$\begin{matrix} a_{k+1} \\ a_k \end{matrix}$	$\cdots$	$\begin{matrix} a_{2k} \\ a_1 \end{matrix}$	a_{2k+1}	$\cdots$

Fig. 1.

During an arbitrary run of T the actions of the Turing machine may be separated into two parts – the management of the input a_i, $i \in N$, in the folded infinite right part of the tape and the computations that finally yield the symbols of the output $T(w)$ in the initial part of the tape after the left boundary marker Z according to the information derived from the input. Whenever a new square for the computations is needed at the end of the work-space in the initial segment of the tape, the input part of the tape shall be folded forwards again, T using the 'folding programme' we are going to describe in the following lines.

Let Γ' be a proper subset of Γ with $\Sigma \subseteq \Gamma'$, and let $\binom{x}{y} \in \Gamma - \Gamma'$ for any pair of symbols $x, y \in \Gamma'$. Then T can be provided with work-space of arbitrary length for its computations by using the following algorithm:

(i) At the beginning of any run, T produces a tape of the form $Zb^l\binom{a_1}{a_{l+1}} \cdots \binom{a_l}{a_{2l}} a_{2l+1} \cdots$, where l is an integer only depending on T and $b \in \Gamma - \Gamma'$ is a blank symbol as is used in the finite case.

(ii) If (and only if whenever (strongly) deterministic Turing machines are considered) T needs a new square of the tape for some computations, the *folding programme* is called, which only acts on the input part of the tape, but neither changes any symbol of the actual work-space nor goes back to any symbol of $T(w)$ already computed in the initial segment of the tape: Starting in a configuration $ZX_1 \cdots X_k q \binom{y_1}{y_{k+1}} \cdots \binom{y_k}{y_{2k}} y_{2k+1} \cdots$ with $q \in K$, $X_i \in \Gamma$, $1 \leq i \leq k$, $y_j \in \Gamma'$, $1 \leq j \leq m$, $y_j = a_j$, $j \geq m$, for some $k, m \in N$ and $T_\Sigma(w)/n = \prod_{i=1}^{n} X_i$ for some $n \leq k$, T folds

the input part of the tape forwards by shifting the information concerning $y_1, \ldots, y_k$ two squares to the right and by packing y_{k+1} together with y_{k+2}, and finishes the programme in the configuration

$$ZX_1 \cdots X_k qb\binom{y_{k+2}}{y_{k+1}}\binom{y_{k+3}}{y_k} \cdots \binom{y_{2k}}{y_3}\binom{y_{2k+1}}{y_2}\binom{y_{2k+2}}{y_1} y_{2k+3} \cdots .$$

By these actions T can be provided with sufficient work-space during an arbitrary run without losing any information concerning the input initially written on the tape. □

The next theorem shows that for real Turing functions the choice of the base k for the expansions of the real numbers in the corresponding consistent functions makes no difference.

Theorem 2. *Let $f:[0,1]\to[0,1]$ be a real function in* TF(DTF, SDTF). *Then for any $k \geq 2$ there exists a (deterministic, strongly deterministic) Turing machine $T^{(k)}$ so that $T_{N_k}^{(k)r}$ equals f.*

Proof. By definition, there is an $l \geq 2$ with $T_{N_l}^{(l)r} = f$. Therefore it is sufficient to construct an equivalent Turing machine $T^{(k)}$ for any other $k \geq 2$. As is well known, a simple algorithm exists for converting the expansion $x^{(k)}$ of the real number x at the base k into an expansion $x^{(l)}$ of x at the base l or vice versa. So $T^{(k)}$ uses this algorithm to compute an arbitrarily long finite part of the expansion at the base l from the given infinite expansion at the base k (only a finite part of computable length of $x^{(k)}$ is used at each time) while simulating $T^{(l)}$. For the opposite conversion, the only difficulty is to determine when the nth digit of $T^{(l)}(x^{(l)})$ can be used, which problem is settled by definition for strongly deterministic Turing machines, whereas for non-deterministic Turing machines a non-deterministic choice is possible. So far as deterministic Turing machines are concerned, the following algorithm has to be applied:

Whenever the simulation of $T^{(l)}$ by $T^{(k)}$ has yielded a new digit $a_m \in N_l$, $T^{(k)}$ assumes that a_m equals $T^{(l)}(x^{(l)})(m)$, $m \in N$, and will never be altered again when the simulation is continued afterwards. $T^{(k)}$ stores the actual work-space, yet leaving the digits already computed of the expansion of $T^{(l)}(x^{(l)})$ at the base k untouched, and calls the conversion algorithm to decide whether a new digit of this expansion of $T^{(l)}(x^{(l)})$ at the base k can be computed by means of the new digit a_m. If a digit a_m that has been assumed to be definite is altered or reached again during the simulation of $T^{(l)}$ by $T^{(k)}$, the original work-space is regained, the simulation continuing where it had stopped to try to gain a new digit of $T^{(k)}(x^{(k)})$, i.e. of the expansion of $T^{(l)}(x^{(l)})$ at the base k. A complete non-oscillating run of $T^{(k)}$ therefore yields an ω-word $T_{N_k}^{(k)}(x^{(k)})$ iff $T_{N_l}^{(l)}(x^{(l)})$ exists, and moreover $\mu_k(T^{(k)}(x^{(k)})) = \mu_l(T^{(l)}(x^{(l)})) = f(x)$.

Therefore we can conclude $T_{N_k}^{(k)r} = T_{N_l}^{(l)r} = f$ for deterministic Turing machines, too, which completes the proof. □

Because of Theorem 2 we may restrict our attention to the input alphabet $\Sigma = N_2$ so far as real functions (in SDTF, DTF or TF) and real numbers (in SDTN, DTN or TN) are concerned.

The following lemma is an immediate consequence of the fact that the computation of a Turing number does not depend on the input.

Lemma 2. *Let x be a real number. Then x is an element of* TN(DTN, SDTN) *iff there is a (deterministic, strongly deterministic) Turing machine $T = (K, N_2, \Gamma_Z, Z, \delta, q_0)$ so that $Zq_0b^\omega \vdash_T^\omega Zx^{(2)}$, where $x^{(2)}$ is an expansion of x at the base 2 and $b \in \Gamma\text{-}\Sigma$ is the blank symbol used when regarding the acceptance of finite words.*

By Lemma 2, another definition of real numbers which can be computed by Turing machines is gained, and for strongly deterministic Turing numbers this definition was already proposed in [9].

Whereas the inclusions DTN $\subseteq$ TN and DTF $\subseteq$ TF are obvious by the definitions, the following theorem shows that these inclusions are proper.

Theorem 3. DTN $\subsetneq$ TN.

Proof. Let $\{T_i\}_{i=1}$ be an effective enumeration of all deterministic Turing machines with the input alphabet $\Sigma = N_2$, and let $R_D = \prod_{i=1}^{\infty} a_i \in N_2^\omega$ be defined by $a_i = 1$ if $T_i(b^\omega)(i) = 0$ and $a_i = 0$ otherwise. By diagonalization, R_D is not in DTN, but a non-deterministic Turing machine T with $T(b^\omega) = R_D$ can be described as follows:

Successively T generates an encoding of the Turing machine $T_i = (K_i, N_2, \Gamma_{Z_i}^{(i)}, Z_i, \delta_i, q_i)$ and non-deterministically guesses the result of the run of T_i on b^ω. If T_i is supposed to halt after a finite number of steps, T simulates T_i on b^ω until T_i halts, and afterwards T_i need not be considered any more. If an oscillating run (complete or not) of T_i on b^ω is guessed, T non-deterministically chooses a square of the tape of T_i to which T_i will return infinitely many times and marks it. If T_i is supposed to have a complete non-oscillating run so that $T_i(b^\omega) = w_i \in \Gamma^{(i)\omega}$, T simulates T_i until w_i/i is guessed to have been computed definitely. The ith digit of R_D now can be added – according to the above definition of a_i – to the sequence $a_1 \cdots a_{i-1}$ already computed. After these actions T simulates the Turing machines T_j, $1 \leq j \leq i$, until each oscillating T_j has reached its marked square again, and until each non-oscillating T_j may be assumed to have computed $w_j(i)$ definitely, T again non-deterministically choosing the right moment. Then T begins the next cycle of this algorithm by generating the encoding of T_{i+1}. Obviously, any complete non-oscillating run of T on b^ω yields R_D, and by Lemma 2 we conclude $\mu_2(R_D) \in$ TN-DTN. □

Corollary 1. DTF $\subsetneq$ TF.

Proof. By Theorem 3, the real function $f(x) = \mu_2(R_D)$ for all $x \in [0, 1]$ is an element of TF-DTF. □

Thus, non-deterministic Turing machines are strictly more powerful in defining real functions and numbers than deterministic Turing machines, which in turn are strictly more powerful than strongly deterministic Turing machines, which shall be proved in the second part of this section.

The next theorem reveals an important topological property of strongly deterministic Turing functions.

Theorem 4. *Each function f in* SDTF *is continuous on any closed interval* $[a, b] \subseteq \mathrm{dom}(f)$.

Proof. Let $\varepsilon \in (0, 1)$, $x \in [a, b]$, $x^{(k)}$ be the infinite expansion of x at the base k, and let $T = (K, N_k, \Gamma_Z, Z, \delta, q_0)$ be a strongly deterministic Turing machine so that $T^{\mathrm{r}}_{N_k} = f$. Then for any $n \in N$ with $k^{-n} < \varepsilon$, by the definition of complete non-oscillating runs of strongly deterministic Turing machines, there must be an $m \in N$ so that T only needs the information about $x^{(k)}/m$ to compute the first n digits of $f(x)$. As T is strongly deterministic, for all $y^{(k)} \in N^{\omega}_k$, $y^{(k)}/m = x^{(k)}/m$ implies $T(y^{(k)})/n = T(x^{(k)})/n$ whenever $T(y^{(k)})$ exists. Therefore we conclude that $T^{\mathrm{r}}_{N_k}$ must be continuous on any closed interval of $\mathrm{dom}(T^{\mathrm{r}}_{N_k})$, being aware of the fact that the consistence of T_{N_k} implies the continuity of $T^{\mathrm{r}}_{N_k}$ in the critical rational points with two expansions in $Q(k)$. □

Example 4. Let $x_0, x_1, \ldots, x_n$, $0 = x_0 < x_1 < \cdots < x_n = 1$, and $y_0, y_1, \ldots, y_n$ be strongly deterministic Turing numbers, $n \geq 2$, and let f be the real function defined by $f(x) = y_i$ for all $x \in [x_{i-1}, x_i] - \{x_{i-1}\}$, $1 \leq i \leq n$, $f(0) = y_0$. By Theorem 4, f is an element of DTF-SDTF, because f is not continuous on $[0, 1] = \mathrm{dom}(f)$. The construction of a deterministic Turing machine inducing f is straightforward and therefore omitted (see e.g. Example 3).

Yet, as Coroliary 2 will show, the fact that a real function $f \in$ DTF is continuous, is not sufficient to guarantee that f is a strongly deterministic Turing function.

The following notion enables us to give a rather simple example of a deterministic, but not strongly deterministic Turing number.

Definition 6. For any alphabet Σ and any ω-word $x \in \Sigma^{\omega}$ let $\mathrm{Init}(x)$ be the set of all prefixes x/i, $i \in N_0$.

Lemma 3. *Let $x \in N^{\omega}_k$, $k \geq 2$; then $\mu_k(x) \in$* SDTN *iff* $\mathrm{Init}(x)$ *is a recursive set.*

Proof. Let T be a strongly deterministic Turing machine with $T(b^{\omega}) = x$; a deterministic Turing machine T' so that $L(T') = \mathrm{Init}(x)$ and T' halts on any input $y \in N^*_k$ can be described as follows: T' simulates T until $x/|y|$ has been computed, and compares $x/|y|$ with y. If $x/|y| = y$, T' enters a final state, otherwise T' halts entering a special non-final state.

To prove the opposite direction, let T' be a deterministic Turing machine halting on any input $y \in N_k^*$ so that $L(T') = \text{Init}(x)$. Then a strongly deterministic Turing machine T with $T(b^\omega) = x$ can be constructed as follows: Having already computed the first $n \in N$ digits of x, T simulates T' on all words $(x/n)y$, $y \in N_k$. By definition, T' halts on each of these words, and there is exactly one y_0 in N_k so that T' accepts $(x/n)y_0$ by entering a final state. As y_0 must equal $x(n+1)$, T can add this new digit to the initial segment of its tape where the digits of x are listed successively, and continue with simulating T' on all words $(x/(n+1))y$, $y \in N_k$, etc. Obviously, T is a strongly deterministic Turing machine which successively computes the digits $x(i)$, i.e. $T(b^\omega) = x$. □

As the preceding lemma shows, a strongly deterministic Turing number can be defined without using the notions and the theory of ω-words, which is not possible in the case of a deterministic or arbitrary Turing number, where in general no effective algorithm need exist which for any $n \in N$ computes the nth digit of an expansion of such a real number at a base k. But by an obvious modification of the proof of Lemma 3 we immediately obtain the following characterization of a strongly deterministic Turing number, which was also used in [9, Chapter 9], besides the characterization that was proposed following Lemma 2, to get real numbers which can be defined by Turing machines: For any $\mu_k(x) \in \text{SDTN}$, $x \in N_k^\omega$, $k \geq 2$, there exists a deterministic Turing machine which for any $n \in N$, when started with n on its tape, computes $x(n)$ and halts in a final state.

The characterization of strongly deterministic Turing numbers by recursive sets and the famous halting problem for Turing machines are the basic tools for proving the following theorem.

Theorem 5. SDTN$\subsetneq$DTN.

Proof. Let $\{T_i\}_{i=1}$ be an effective enumeration of all deterministic Turing machines with the input alphabet $\Sigma = N_2$, and let $R_U = \prod_{i=1}^{\infty} a_i \in N_2^\omega$ be defined by $a_i = 1$ if T_i halts on b^ω and $a_i = 0$ otherwise. If R_U defined a strongly deterministic Turing number, Init(R_U) would be a recursive set so that it would be sufficient to check all words $v \in N_2^*$ with $|v| = i$ for deciding the question whether T_i halts on b^ω, which leads to a contradiction. A deterministic Turing machine T with $T(b^\omega) = R_U$ can be described as follows: For any $i \in N$, T successively generates an encoding of T_i and assumes T_i not to halt on b^ω. Then for all $n < i$ one single step of T_n on b^ω is simulated. If T_n halts, the nth digit of R_U first assumed to be 0 is altered to 1. As for any $i \in N$ there is an $m \geq i$ so that for all $k \leq i$ T_k has halted after m steps if it ever does so, a complete non-oscillating run of T on b^ω exists yielding R_U. □

Corollary 2. *There exists a deterministic, but not strongly deterministic Turing function which is continuous on* $[0, 1]$.

Proof. By Theorem 5, the constant function $f(x) = \mu_2(R_U)$ for all $x \in [0, 1]$, which clearly is continuous on $[0, 1]$, is an element of DTF-SDTF. □

The following theorem is a conclusion of the most important results obtained so far in this section.

Theorem 6 (Hierarchy of Turing functions and numbers).

$$\text{SDTF} \subsetneq \text{DTF} \subsetneq \text{TF}, \qquad \text{SDTN} \subsetneq \text{DTN} \subsetneq \text{TN}.$$

To finish this section we are going to reveal an interesting closure property of the families of Turing functions SDTF, DTF, and TF. For this purpose let the composition of two arbitrary real functions f and g, $f \circ g$, be defined by $(f \circ g)(x) = f(g(x))$ for any real number x.

Theorem 7. *The families of Turing functions* SDTF, DTF, *and* TF *are closed under composition.*

Proof. Let f and g be two ((strongly) deterministic) Turing functions and $T^{(f)}$ and $T^{(g)}$ be ((strongly) deterministic) Turing machines with $T^{(f)r}_{N_k} = f$ and $T^{(g)r}_{N_k} = g$ for some $k \geq 2$. Then a ((strongly) deterministic) Turing machine $T^{(f \circ g)}$ with $T^{(f \circ g)r}_{N_k} = f \circ g$ can be constructed as follows:

(i) Let $T^{(f)}$ and $T^{(g)}$ be strongly deterministic Turing machines. Started with $w \in N_k^\omega$ on its tape, $T^{(f \circ g)}$ simulates $T^{(g)}$ until the first digit of $T^{(g)}(w)$ has been computed, and continues with simulating $T^{(f)}$. Whenever $T^{(f)}$ needs a new digit of $T^{(g)}(w)$, $T^{(f \circ g)}$ interrupts the simulation of $T^{(f)}$ to compute the next digit of $T^{(g)}(w)$ by simulating $T^{(g)}$ on the original input w, whereafter $T^{(f \circ g)}$ continues the simulation of $T^{(f)}$ on the already computed prefix of $T^{(g)}(w)$. Obviously, $T^{(f \circ g)}$ is a strongly deterministic Turing machine as are $T^{(f)}$ and $T^{(g)}$ with $T^{(f \circ g)r}_{N_k} = T^{(f)r}_{N_k} \circ T^{(g)r}_{N_k}$.

(ii) For deterministic Turing machines $T^{(f)}$ and $T^{(g)}$ we adopt an idea already used in the proof of Theorem 2 as well as the algorithm explained in (i): Whenever $T^{(f)}$ needs a new digit of $T^{(g)}(w)$, $T^{(f \circ g)}$ simulates $T^{(g)}$ on w until the next symbol $a_l \in N_k$ has been computed, which is assumed to be the next digit of $T^{(g)}(w)$ and therefore never to be altered again, and stores the actual work-space, continuing with the simulation of $T^{(f)}$ on a word $\prod_{i=1}^{l} a_i$, $a_i \in N_k$, $1 \leq i \leq l$, $l \in N$, which is assumed to equal the prefix $T^{(g)}(w)/l$. If such a digit a_l that has been assumed to be definite is reached or altered during a simulation of $T^{(g)}$ by $T^{(f \circ g)}$, the original work-space is regained, and $T^{(f \circ g)}$ simulates one more step of $T^{(g)}$ on w before it stores the actual work-space and continues with simulating $T^{(f)}$ on $(\prod_{i=1}^{l-1} a_i)a'_l$, where $a'_l \in N_k$ may differ from the digit a_l computed before. A complete non-oscillating run of the deterministic Turing machine $T^{(f \circ g)}$ yields an ω-word $T^{(f \circ g)}(w) \in N_k^\omega$ iff $T^{(g)}(w) \in N_k^\omega$ as well as $T^{(f)}(T^{(g)}(w)) \in N_k^\omega$, and by construction $T^{(f \circ g)}(w) = T^{(f)}(T^{(g)}(w))$ for all $w \in N_k^\omega$, i.e. $T^{(f \circ g)r}_{N_k} = f \circ g$.

(iii) If $T^{(f)}$ and $T^{(g)}$ are non-deterministic Turing machines, $T^{(f\circ g)}$ computes $T^{(f\circ g)}(w) = T^{(f)}(T^{(g)}(w))$, $w \in N_k^\omega$, by using a similar algorithm like that described in (i), but has to decide in a non-deterministic way when the simulation of $T^{(g)}$ on w has yielded a new digit of $T^{(g)}(w)$ that it can use unrestrictedly during the simulation of $T^{(f)}$. □

The following corollary is an immediate consequence of the fact that any Turing number can be regarded as a special Turing function.

Corollary 3. *Let* $f \in$ SDTF (*resp.* DTF, TF) *and* $x \in$ SDTN (*resp.* DTN, TN); *then* $f(x) \in$ SDTN (*resp.* DTN, TN), *too.*

An interesting subfamily of SDTF, which is also closed under composition, is the family of real functions which can be obtained from consistent functions induced by sequential machines that we have already mentioned to be strongly deterministic Turing machines satisfying rather rigorous restrictions. According to Proposition 4.2 in [6, p. 370], any real function which is induced by a sequential machine maps rational numbers into rational numbers. Hence, the family of real numbers that can be obtained from constant consistent functions induced by sequential machines equals Q, and for any rational number q also $f(q)$ is a rational number for any real function f which is induced by a sequential machine. As the results of the following sections will show, Q is properly included in SDTN, which may also be deduced from Lemma 3 and the corresponding characterization of rational expansions by regular sets, because an obvious construction shows that for any $x \in N_k^\omega$, $k \geq 2$, $\mu_k(x) \in Q(k)$ iff Init(x) is a regular set.

4. Transcendent strongly deterministic Turing numbers

In this section the notion of the relative frequency of symbols 1 in words over the alphabet N_2 is used to define special strongly deterministic Turing numbers which can be proved to be transcendent real numbers.

Definition 7. For any $y \in N_2^+$ let $\rho(y) = n_1(y)/(n_1(y)+n_0(y))$; for any $x \in N_2^\omega$ let $\rho_n(x) = \rho(x/n)$ for all $n \in N$, and $\rho(x) = \lim_{n\to\infty} \rho_n(x)$.

The number $\rho(x)$ need not exist, because the numbers $\rho_n(x)$ may oscillate so that the sequence of the $\rho_n(x)$ cannot be convergent. As for any rational number $x \in Q$ the non-terminating expansion $x_1 \in Q(2)$ yields $\rho(x_1) = 1$ whereas the terminating expansion $x_2 \in Q(2)$ yields $\rho(x_2) = 0$, the function $\rho: N_2^\omega \to N_2^\omega$ is not consistent; inspite of this, $\rho \in \mathrm{TF}(N_2)$ as is shown in the following theorem.

Theorem 8. *There exists a Turing machine* T *so that for all* $w \in N_2^\omega$, $\mu_2(T_{N_2}(w)) = \rho(w)$ *iff* $\lim_{n\to\infty} \rho_n(w)$ *exists and* $T_{N_2}(w)$ *remains undefined otherwise.*

Proof. Let T have already computed the digits of $\rho(w)/n$, $n \in N_0$, using $\rho_k(w)$, $1 \leq k < m$ for an $m \in N$. Now T computes $\rho_m(w)/n$ and the natural numbers

$$x_m = \sum_{i=0}^{n-1} (\rho_m(w))(n-i)2^i \quad \text{and} \quad y_m = \sum_{j=0}^{n-1} (\rho(w)/n)(n-j)2^j,$$

only continuing its actions when $|x_m - y_m| \leq 1$. Before regarding $\rho_{m+1}(w)$, T nondeterministically may guess the next $l \geq 1$ digits $(\rho(w))(n+1), \ldots, (\rho(w))(n+l)$, the condition $(\rho(w))(n+j) = 1$ for some $j \leq l$ keeping T from computing terminating expansions of some elements of $Q(2)$. □

Similar constructions as the following one will also be used in the sequel.

Lemma 4. *For any real number $x \in [0, 1]$ there is an ω-word $w \in N_2^\omega$ with $\rho(w) = x$.*

Proof. The ω-word $w = \prod_{i=1}^{\infty} a_i$ can be defined recursively by $a_1 = 1$, $a_{n+1} = 0$ if $\rho(\prod_{i=1}^{n} a_i) \geq x$ and $a_{n+1} = 1$ otherwise for all $n \in N$. Obviously, $\lim_{n \to \infty} \rho_n(w) = x$. □

Before establishing a class of transcendent strongly deterministic Turing numbers, another class of irrational strongly deterministic Turing numbers is considered in the following example.

Example 5. Let b_1, $b_2 \in \mathrm{SDTN}$, $0 < b_1 < b_2 < 1$, and let $z(b_1, b_2) = 10^{l_1}1^{m_1}0^{l_2}1^{m_2} \cdots$ be defined by choosing l_k and m_k the smallest natural numbers so that $\rho(1 \cdots 0^{l_k}) < b_1$ and $\rho(1 \cdots 0^{l_k}1^{m_k}) > b_2$ for all $k \geq 1$; then $\mu_2(z(b_1, b_2)) \in \mathrm{SDTN}\text{-}Q$:

Because of the recursive definition of the numbers l_k and m_k, the digits of $z(b_1, b_2)$ obviously can be computed consecutively by a strongly deterministic Turing machine, provided T can check $x < y$ (resp. $x > y$) for any $x \in Q$ and any $y \in \mathrm{SDTN}$: Let p, q be natural numbers so that $p/q = x$. If $y \in Q$, let p', q' be natural numbers so that $p'/q' = y$. Then pq' and $p'q$ are compared instead of x and y. If $y \in \mathrm{SDTN}\text{-}Q$, the expansions of x and y must differ in some digit. As for some $n \in N$ the sequences $l_n, l_{n+1}, \ldots$ and $m_n, m_{n+1}, \ldots$ must be strictly increasing, $z(b_1, b_2) \notin Q(2)$.

Definition 8. Let $f : N_0 \to N$ be a total recursive function. For any $b \in \mathrm{SDTN}$, $b > 0$, define $z(b, f) = 10^{l_1}1^{m_1}0^{l_2}1^{m_2} \cdots$ by choosing $l_0 = 0$, $m_0 = 1$, and, for all $k \geq 1$, $l_k = f(k-1)\sum_{i=0}^{k-1}(l_i + m_i)$ as well as m_k to be the smallest natural number so that $\rho(1 \cdots 0^{l_k}1^{m_k}) > b$ for $b < 1$ resp. $\rho(1 \cdots 0^{l_k}1^{m_k}) = 1 - 1/(k+2)$ for $b = 1$.

The above definitions will be justified in the following lemma, where the real numbers $\mu_2(z(b, f))$ are proved to be strongly deterministic Turing numbers like those considered in Example 5:

Lemma 5. *For any total recursive function $f : N_0 \to N$ and any strongly deterministic Turing number $b > 0$, $\mu_2(z(b, f)) \in \mathrm{SDTN}$.*

Proof. Using similar arguments like in Example 5, the statement can be proved for all $b<1$. For $b=1$, let $L_k=\sum_{i=0}^{k} l_i$ and $M_k=\sum_{i=0}^{k} m_i$ for all $k\geq 0$. Thus, $l_{k+1}=f(k)(L_k+M_k)$ and $\rho(1\cdots 0^{l_k}1^{m_k})=M_k/(L_k+M_k)$, which yields a recursive definition for m_k: $\rho(1\cdots 0^{l_k}1^{m_k})=(k+1)/(k+2)$ implies $M_k=(k+1)L_k$, therefore $m_{k+1}=M_{k+1}-M_k=(k+2)(L_k+l_{k+1})-(k+1)L_k=L_k+(k+2)f(k)(L_k+M_k)$. Thus, $m_k\in N$ is well-defined for all $k\geq 0$, and we may conclude $\mu_2(z(1,f))\in$ SDTN. □

We now are going to prove the real numbers $\mu_2(z(b,f))$ to be transcendent, which first requires an exact definition of this notion:

Definition 9. A real number x is called *algebraic*, if there is a real polynomial $p(z)=\sum_{i=0}^{n} a_i z^i$, $n\in N$, $|a_i|\in N_0$ for $0\leq i\leq n$, $a_n>0$, so that x is a *root* of $p(z)$, i.e. $p(x)=0$. If x is not algebraic, x is called *transcendent.*

The following theorem is taken from [7] and can be used to easily proving certain real numbers to be transcendent.

Theorem 9. *Let $x>0$ be a real number so that for any $n\in N$ there are infinitely many different natural numbers p, q with $(p,q)=1$ (where (p,q) denotes the greatest common divisor of p and q) and $|x-p/q|<C/q^n$, where C is a real number only depending on x; then x is transcendent.*

Example 6. By applying Theorem 9, the real number $\mu_2(\prod_{i=1}^{\infty} 10^{i!})=\mu_2(10^1 10^2 10^6 10^{24} 10^{120} 10^{720}\cdots)$ is easily shown to be a transcendent strongly deterministic Turing number, where we define $0!=1$ and $n!=(n-1)!n$ for all $n\geq 1$.

In the following lemma a class of real numbers, which properly contains a subclass of the real numbers $\mu_2(z(b,f))$, is shown to consist of transcendent numbers only.

Lemma 6. *Any real number $\mu_2(x)$, $x=10^{l_1}1^{m_1}0^{l_2}1^{m_2}\cdots$ with $l_k, m_k\in N$ and $l_k\geq (k-1)\sum_{i=0}^{k-1}(l_i+m_i)$ for all $k\geq 1$, is transcendent.*

Proof. Let L_k and M_k be defined like in Lemma 5. By Theorem 9 we only have to show that, for any $n\in N$, $|\mu_2(x)-p_k/q_k|<2^{-n(L_k+M_k)}$ holds true for all $k\geq n$, p_k and q_k being defined by

$$p_k=\sum_{i=0}^{L_k+M_k-1} x(L_k+M_k-i)2^i \quad \text{and} \quad q_k=2^{(L_k+M_k)}.$$

Obviously,

$$\begin{aligned}
&\mu_2(x)-p_k/q_k\\
&=\mu_2(1\cdots 0^{l_k}1^{m_k}0^{l_{k+1}}1^{m_{k+1}}\cdots)-\mu_2(1\cdots 0^{l_k}1^{m_k}0^{\omega})\\
&=\mu_2(1\cdots 0^{l_k}1^{m_k}0^{l_{k+1}}1^{\omega})-\mu_2(1\cdots 0^{l_k}1^{m_k}0^{\omega})\\
&=2^{-(L_k+M_k+l_{k+1})}
\end{aligned}$$

and

$$L_k + M_k + l_{k+1} \geq L_k + M_k + k(L_k + M_k) = (k+1)(L_k + M_k) > n(L_k + M_k).$$

Therefore we conclude

$$|\mu_2(x) - p_k/q_k| < 2^{-(L_k + M_k + l_{k+1})} < 2^{-n(L_k + M_k)} = 1/q_k^n$$

for all $k \geq n$, which completes the proof. □

A class of transcendent strongly deterministic Turing numbers now is easily established:

Theorem 10. *For any $b \in$ SDTN, $b > 0$, and any total recursive function $f: N_0 \to N$ with $f(k) \geq k$ for all $k \geq 1$, the real number $\mu_2(z(b, f)) \in$ SDTN is transcendent. Moreover, the set of all $\rho_n(z(b, f))$, $n \in N$, is dense in $(0, b)$.*

Proof. By Lemma 6, the numbers $\mu_2(z(b, f))$ are transcendent. For the second sentence, observe that $\lim_{k\to\infty} \rho(z(b, f)/(L_k + M_{k-1})) = 0$ and $\lim_{k\to\infty} \rho(z(b, f)/(L_k + M_k)) = b$, which implies that $\rho(z(b, f)/n)$ oscillates between 0 and b, the differences $|\rho(z(b, f)/n) - \rho(z(b, f)/(n-1))|$ becoming arbitrarily small. □

For an expansion $z(1, f) = 10^{l_1}1^{m_1}0^{l_2}1^{m_2} \cdots$ the numbers l_k and m_k are given by recursions so that for a suitable function f they may be described by explicit formulas as in the following example.

Example 7. Let $f(0) = 1$ and $f(k) = k$ for all $k \geq 1$, and observe the expansion $z(1, f)$. Using the common notation, the equations $l_{k+1} = L_{k+1} - L_k$, $l_{k+1} = k(L_k + M_k)$, and $M_k/(L_k + M_k) = 1 - 1/(k+2)$ yield the recursion $L_1 = 1$, $L_{k+1} = (k+1)^2 L_k$ for all $k \geq 1$, which has the solution $L_k = (k!)^2$, $k \geq 1$. Then the formulas $M_k = (k+1)!k!$, $m_k = (k-1)!k!(k^2 + k - 1)$, and $l_1 = 1$, $l_{k+1} = (k!)^2(k^2 + 2k)$ for all $k \geq 1$ are immediately derived from the formula for L_k ($z(1, f) = 1010001^{10}0^{32}1^{132}0^{540}1^{2736}0^{13824} \cdots$).

5. Transcendent non-strongly deterministic Turing numbers

In this section all Turing numbers which are not strongly deterministic are shown to be transcendent by proving all algebraic numbers to be strongly deterministic Turing numbers whereas, by Example 6, the converse of this sentence does not hold true.

Lemma 7. *Let $p(z) = \sum_{i=0}^{n} a_i z^i$, $n \in N$, $|a_i| \in N_0$, $a_n > 0$, be a real polynomial; then any irrational root $x \in [0, 1]$ of $p(z)$ is a strongly deterministic Turing number.*

Proof. For any real number y let $\text{sign}(y)$ be defined by $\text{sign}(0)=0$, $\text{sign}(y)=1$ for all $y>0$ and $\text{sign}(y)=-1$ for all $y<0$. Let $p'(z)=\sum_{i=1}^{n} i a_i z^{i-1}$ denote the first derivate of $p(z)$ and let q_0, $m \in N$ so that x is the only root of $p(z)$ in the interval $[q_0 2^{-m}, (q_0+1)2^{-m}] \subseteq [0, 1]$ and $p'(z)$ changes its sign from -1 to $+1$ resp. from $+1$ to -1 at most once in this interval. If $\text{sign}(p(q_0 2^{-m})) \neq \text{sign}(p((q_0+1)2^{-m}))$, the algorithm of bisection described in the following lines can be applied directly to $p(z)$, otherwise, if $\text{sign}(p(q_0 2^{-m})) = \text{sign}(p((q_0+1)2^{-m}))$, then x is the only root of $p'(z)$ in the interval $[q_0 2^{-m}, (q_0+1)2^{-m}]$ and $\text{sign}(p'(q_0 2^{-m})) \neq \text{sign}(p'((q_0+1)2^{-m}))$ so that the algorithm has to be applied to $p'(z)$ instead of $p(z)$.

Thus, *w. l. o. g.* we may assume that x is the only root of $p(z)$ in the interval $[q_0 2^{-m}, (q_0+1)2^{-m}]$ and that $\text{sign}(p(q_0 2^{-m})) \neq \text{sign}(p((q_0+1)2^{-m}))$. By using the algorithm of bisection, a sequence of intervals $[q_k 2^{-(m+k)}, (q_k+1)2^{-(m+k)}]$ is constructed so that $x \in (q_k 2^{-(m+k)}, (q_k+1)2^{-(m+k)})$ and each step of the algorithm yields a digit of the expansion of x at the base 2: Given q_k, $k \in N_0$, $p((2q_k+1)2^{-(m+k+1)})>0$ is checked by testing $\sum_{i=0}^{n} a_i (2q_k+1)^i 2^{(n-i)(m+k+1)} > 0$. The next interval $[2q_k 2^{-(m+k+1)}, (2q_k+1)2^{-(m+k+1)}]$ resp. $[(2q_k+1)2^{-(m+k+1)}, (2q_k+2)2^{-(m+k+1)}]$ is chosen according to this test so that the evaluations of the polynomial in the bounds of the interval have different signs. Of course, the steps of this algorithm can be simulated by a deterministic Turing machine which, moreover, is strongly deterministic, because each step yields a new digit. □

As any rational number $x \in Q \subsetneq \text{SDTN}$ is algebraic, the following theorem is an immediate consequence of the preceding lemma.

Theorem 11. *Any algebraic real number $x \in [0, 1]$ is a strongly deterministic Turing number.*

Thus, any non-strongly deterministic Turing number is transcendent:

Corollary 4. *Any real number $x \in \text{TN-SDTN}$ is transcendent.*

Example 8. For each effective enumeration of all deterministic Turing machines with the input alphabet $\Sigma = N_2$ the numbers R_D and R_L, constructed in the proofs of the Theorems 3 and 5, are transcendent.

The following theorem is an obvious generalization of Lemma 7, yielding an algebraic closure property of SDTN.

Theorem 12. *Let $p(z)=\sum_{i=0}^{n} a_i z^i$, $n \in N$, $|a_i| \in \text{SDTN}$, $a_n > 0$, be a real polynomial. Then each real root $x \in [0, 1]$ of $p(z)$ is a strongly deterministic Turing number, too.*

Proof. Let $p'(z)$ and $\text{sign}(y)$ be defined as in the proof of Lemma 7. For all $0 \leq i \leq n$, let b_i be an expansion of $|a_i|$ at the base 2, and let $c_i' = 1$ if $a_i > 0$ and $c_i' = 0$

otherwise, $c_i^- = -1$ if $a_i < 0$ and $c_i^- = 0$ otherwise. Then the algorithm of bisection can also be used to prove the more general theorem: For testing $\operatorname{sign}(p((2q_k + 1)2^{-(m+k+1)}))$ now the integer numbers

$$\sum_{i=0}^{n} \left(\operatorname{sign}(a_i) \sum_{j=1}^{l} b_i(j) 2^{l-j} + c_i^+ \right) (2q_k + 1)^i 2^{(n-i)(m+k+1)}$$

and

$$\sum_{i=0}^{n} \left(\operatorname{sign}(a_i) \sum_{j=1}^{l} b_i(j) 2^{l-j} + c_i^- \right) (2q_k + 1)^i 2^{(n-i)(m+k+1)}$$

are successively computed for $l = n + m + k + 2$, $n + m + k + 3, \ldots$ until both numbers are >0 resp. <0. If $p'(z)$ is used for the algorithm of bisection instead of $p(z)$, then the integer numbers

$$\sum_{i=1}^{n} \left(i \left(\operatorname{sign}(a_i) \sum_{j=1}^{l} b_i(j) 2^{l-j} + c_i^+ \right) \right) (2q_k + 1)^i 2^{(n-i)(m+k+1)}$$

and

$$\sum_{i=1}^{n} \left(i \left(\operatorname{sign}(a_i) \sum_{j=1}^{l} b_i(j) 2^{l-j} + c_i^- \right) \right) (2q_k + 1)^i 2^{(n-i)(m+k+1)}$$

are to be successively computed until both have equal signs. Thus, in any case each step of the algorithm used in the proof of Lemma 7 can be modified accordingly, which yields a strongly deterministic Turing machine for each irrational root $x \in [0, 1]$ of the polynomial $p(z)$ with coefficients in SDTN, too. □

Acknowledgment

The author wishes to thank the referee for his useful suggestions, which resulted in a considerably improved exposition of this paper.

References

[1] R.S. Cohen and A.Y. Gold, Theory of ω-languages: part I: Characterizations of ω-context free languages, *J. Comput. System Sci.* **15** (1977) 169–184.

[2] R.S. Cohen and A.Y. Gold, Theory of ω-languages; part II: A study of various models of ω-type generation and recognition, *J. Comput. System Sci.* **15** (1977) 185–208.

[3] R.S. Cohen and A.Y. Gold, ω-Computations on deterministic pushdown machines, *J. Comput. System Sci.* **16** (1978) 275–300.

[4] R.S. Cohen and A.Y. Gold, ω-Computations on Turing machines, *Theoret. Comput. Sci.* **6** (1978) 1–23.

[5] R.S. Cohen and A.Y. Gold, On the complexity of ω-type Turing acceptors, *Theoret. Comput. Sci.* **10** (1980) 249–272.

[6] S. Eilenberg, *Automata, Languages and Machines, Vol. A* (Academic Press, New York, 1974).

[7] G.H. Hardy and E.M. Wright, *Einführung in die Zahlentheorie* (Oldenbourg, München, 1958).
[8] E.J. Hopcroft and J.D. Ullman, *Formal Languages and their Relation to Automata* (Addison-Wesley, New York, 1969).
[9] M.L. Minsky, *Berechnung: Endliche und unendliche Maschinen* (Verlag Berliner Union, Stuttgart, 1971).
[10] A. Salomaa, *Formal Languages* (Academic Press, New York, 1973).
[11] J. Stoer, *Einführung in die Numerische Mathematik I* (Springer, Heidelberg, 1976).